

**UNIVERSITY OF SALENTO**

**Detailed Analysis of Renewable Energy-Based**

**Desalination System for Maritime Vessels**

**PROFESSORS:**

*Gianfranco PARLANGELI*

**SUPERVISOR:**

*Carlo*

*DE GIORGI*

**STUDENT:**

*Sara FARRIS*

*Danilo GIOVANNICO*

Academic year 2024

Sommario

[1. Abstract 4](#_Toc170764893)

[2. Introduction 4](#_Toc170764894)

[3. System Description 4](#_Toc170764895)

[3.1 Dynamic Equations 5](#_Toc170764896)

[3.1.1 First Case for Each Buoy 6](#_Toc170764897)

[3.1.2 Second Case for Each Buoy 6](#_Toc170764898)

[3.2 Complete Case 6](#_Toc170764899)

[3.3 State of Buoys and Hypotheses 7](#_Toc170764900)

[3.3.1 Case 1 - Insufficient Battery Capacity 7](#_Toc170764901)

[3.3.2 Case 2 - Sufficient Battery Capacity 7](#_Toc170764902)

[3.4 Energy Flow Analysis 8](#_Toc170764903)

[3.5 Considerations for Vessels 9](#_Toc170764904)

[3.5.1 State of the Buoys 10](#_Toc170764905)

[3.5.1 Complete System of Buoys 10](#_Toc170764906)

[3.5.2 Transition Matrix 10](#_Toc170764907)

[3.5.3 State Vector at Time k 11](#_Toc170764908)

[3.5.4 Input Vector 11](#_Toc170764909)

[3.5.5 Matrix Input Vector 11](#_Toc170764910)

[3.5.6 Vector of Constant Terms 12](#_Toc170764911)

[4. Code implementation and results 12](#_Toc170764912)

[4.1 Case n°2 13](#_Toc170764913)

[4.1.1 Results 15](#_Toc170764914)

[4.1.2 General Observations 18](#_Toc170764915)

[4.2 Case n°4 19](#_Toc170764916)

[4.2.1 Results 20](#_Toc170764917)

[4.2.2 General Observations 23](#_Toc170764918)

[4.3 Case n°6 24](#_Toc170764919)

[4.3.1 Results 25](#_Toc170764920)

[4.3.2 General Observations 28](#_Toc170764921)

[4.4 Case n°7 28](#_Toc170764922)

[4.4.1 Results 31](#_Toc170764923)

[4.4.2 General Observations 34](#_Toc170764924)

[5 Request the distribution model between and within the ASes and assign requests 34](#_Toc170764925)

[5.1 Simple method 35](#_Toc170764926)

[5.2 Round Robin method 35](#_Toc170764927)

[5.2.1 Function to assign vessel requests to buoys using round robin 36](#_Toc170764928)

[5.3 Greedy method 37](#_Toc170764929)

[5.3.1 Function to assign vessel requests to buoy systems using an optimized greedy algorithm 39](#_Toc170764930)

[5.4 Problem of the Byzantine generals 40](#_Toc170764931)

[5.4.1 Function to share and verify data between systems thanks to the Byzantine generals problem 43](#_Toc170764932)

[6 Observations and Conclusions 46](#_Toc170764933)

# Abstract

Sea waves are becoming an increasingly important energy source, although still difficult to implement on a large scale. However, small-scale projects are emerging that can change people's lives. One example is a zero-impact desalination system for seawater. The project proposes energy applications derived from waves, not for electricity generation, but for seawater desalination, a crucial issue for many populations in underdeveloped areas with water supply problems. It involves a small but fully energy-autonomous desalination plant. The base is a kind of floating device that captures the energy of sea surface oscillations to pressurize seawater and start the desalination process through reverse osmosis. This process not only produces potable water but also transports it to the mainland through a system of pipes. The system's main advantage is its energy autonomy, which eliminates the need for fossil fuels and reduces costs compared to traditional desalination processes.

# Introduction

This report presents a detailed analysis of a network of buoys designed to produce electricity from renewable sources, which is subsequently used for water desalination. These buoys serve as replenishment points for maritime vessels, providing both desalinated water and energy. The goal is to optimize the management of energy production and water desalination to ensure efficient use of resources.

The increasing need for sustainable and autonomous systems in marine environments has led to the development of buoys equipped with renewable energy sources and desalination capabilities. This report explores the dynamic model of such a system, providing a mathematical framework to describe the accumulation and utilization of energy and water over time.

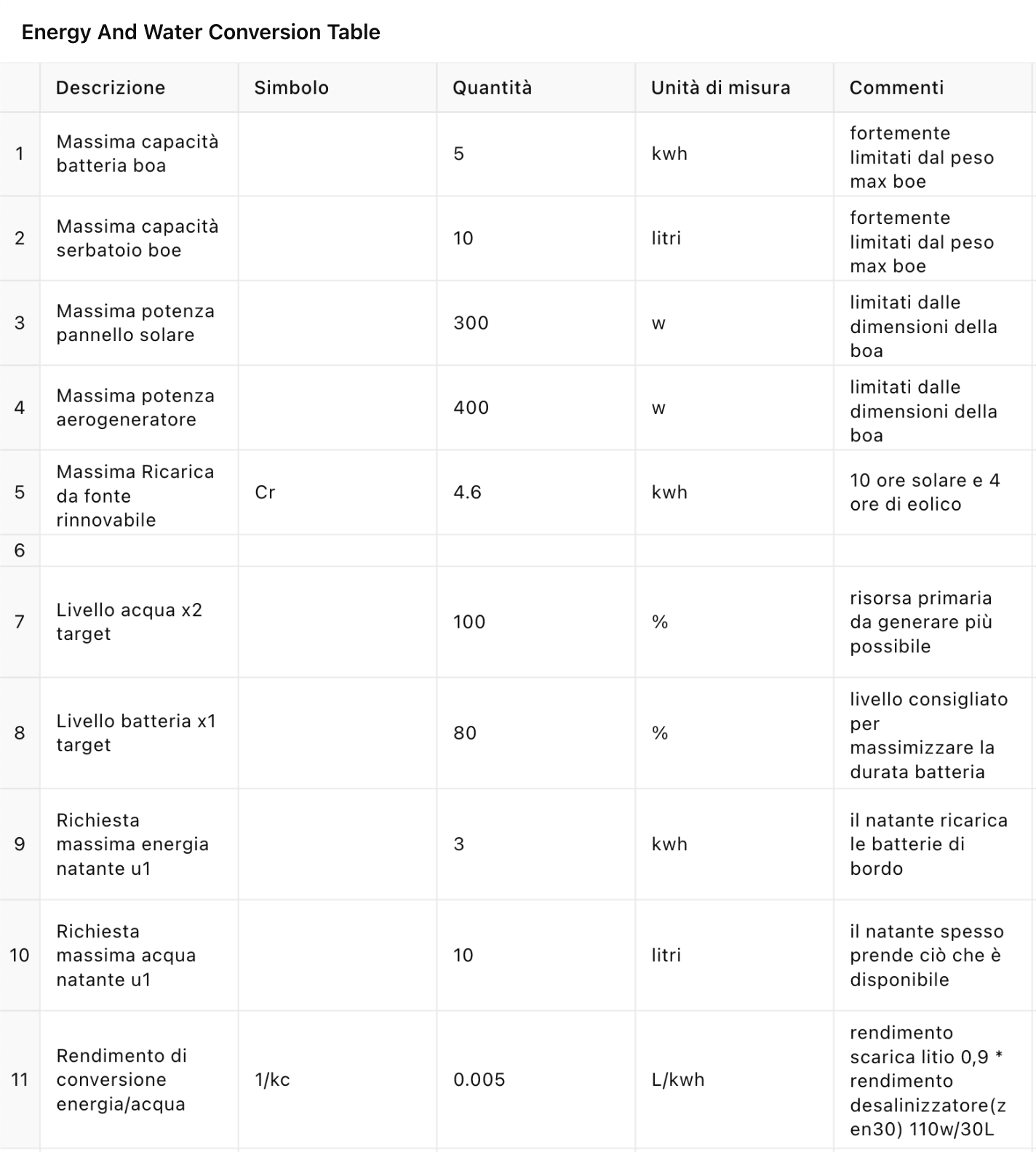
# System Description

Each **buoy** is equipped with:

* **Energy Production System**: Converts renewable energy sources into electricity, denoted as cr​.
* **Battery**: Stores the generated electricity with a discharge efficiency rs​.
* **Desalination System**: Uses stored energy to desalinate water with an efficiency rd​.

The requests from the **vessels** at each instant k occur randomly both in terms of the number of requests and in terms of the maximum resources to be requested (a vessel cannot request a number of resources that falls below the minimum threshold that each buoy can offer ).

Below is the table of specifications for each buoy:

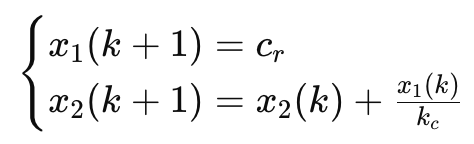


## 3.1 Dynamic Equations

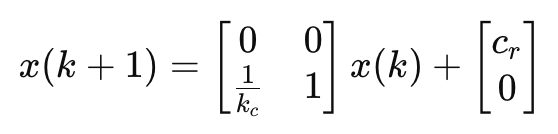
The system's behavior over time is described by a set of dynamic equations, which capture the evolution of accumulated energy and desalinated water.

### 3.1.1 First Case for Each Buoy

The first case considers the basic operation of the buoy without external interference:



In matrix form:

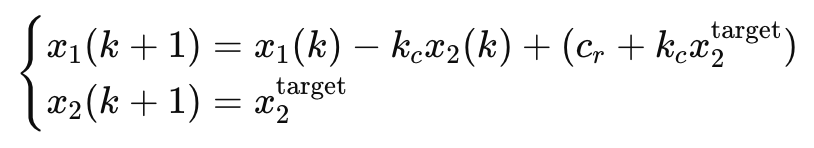


Where:

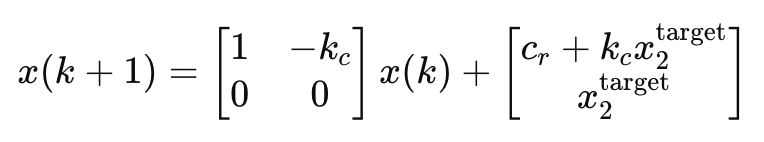
* x1​ represents the accumulated energy.
* x2 represents the accumulated water.
* cr is the constant amount of renewable energy available.
* kc is an energy-to-water conversion parameter.

### 3.1.2 Second Case for Each Buoy

The second case includes target values for water accumulation, introducing a more complex interaction:

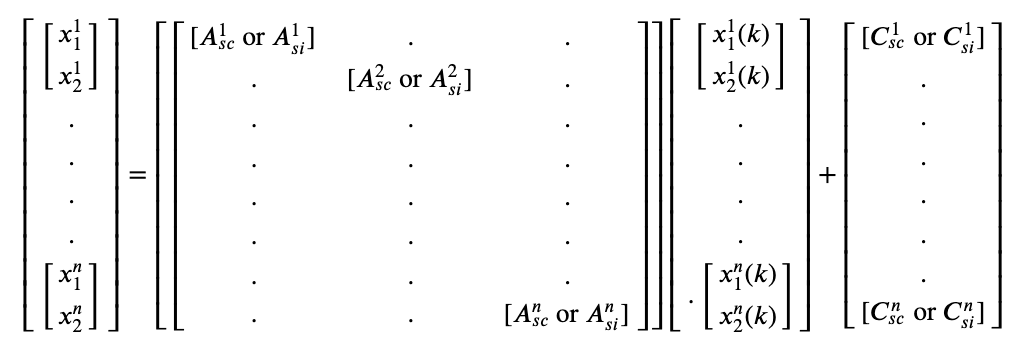


In matrix form:



## 3.2 Complete Case

Combining both scenarios, the complete model accounts for the state of multiple buoys:



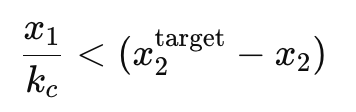
* **Asc**: Complete discharge scenario where all available energy is used.
* **Asi** ​: Incomplete discharge scenario where only part of the available energy is used.
* **Csc, Csi**: Vector of constant terms represents the fixed contributions of energy and water.

## 3.3 State of Buoys and Hypotheses

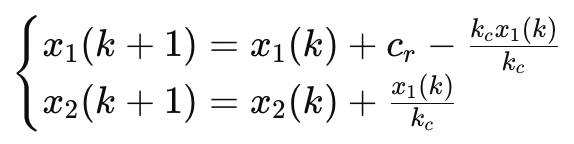
The system prioritizes water production, aiming to reach target values **x1target** and **x2target** ​. Two cases are considered:

### 3.3.1 Case 1 - Insufficient Battery Capacity

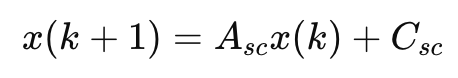
If the battery cannot recharge the entire tank, represented by the inequality:



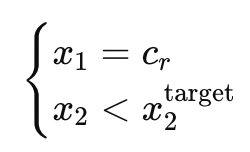
The dynamic equations are:



In matrix form:

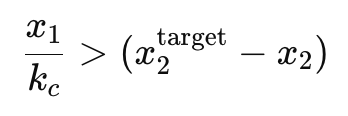


Resulting in:

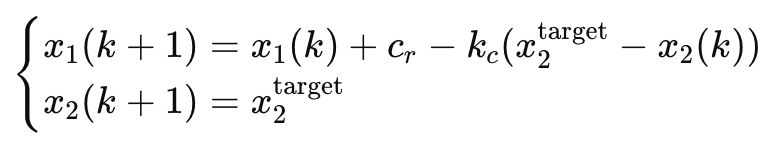


### 3.3.2 Case 2 - Sufficient Battery Capacity

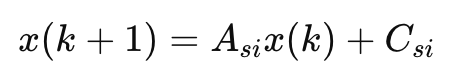
If the battery can recharge the entire tank:



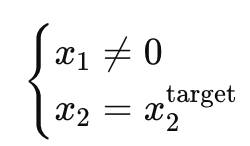
The dynamic equations are:



In matrix form:



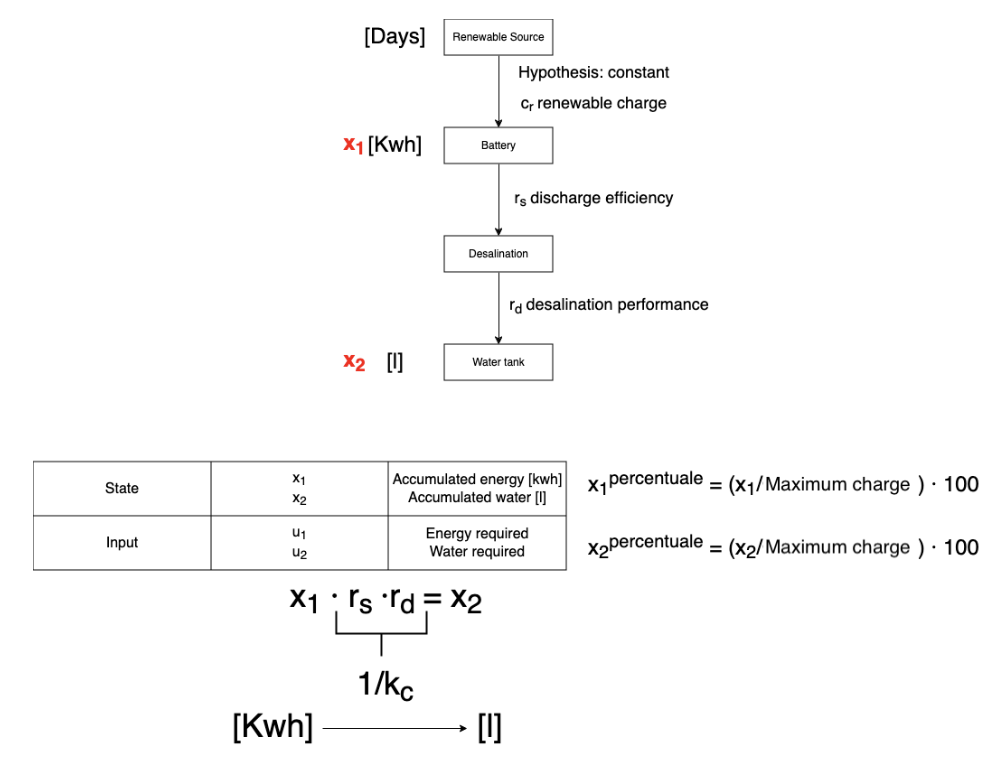
Resulting in:



## 3.4 Energy Flow Analysis

The energy flow diagram illustrates the transformation of renewable energy into water via the battery and desalination processes:

* **State Variables**: x1​ (Accumulated energy in kWh), x2​ (Accumulated water in liters).
* **Input Variables**: u1​ (Energy required), u2​ (Water required).



The energy flow diagram shows the transformation of renewable energy into water through the battery and desalination process:

1. Renewable energy cr.
2. Battery with discharge efficiency rs.
3. Desalination with efficiency rd.
4. Water accumulation in the tank.

Where the conversion is given by:



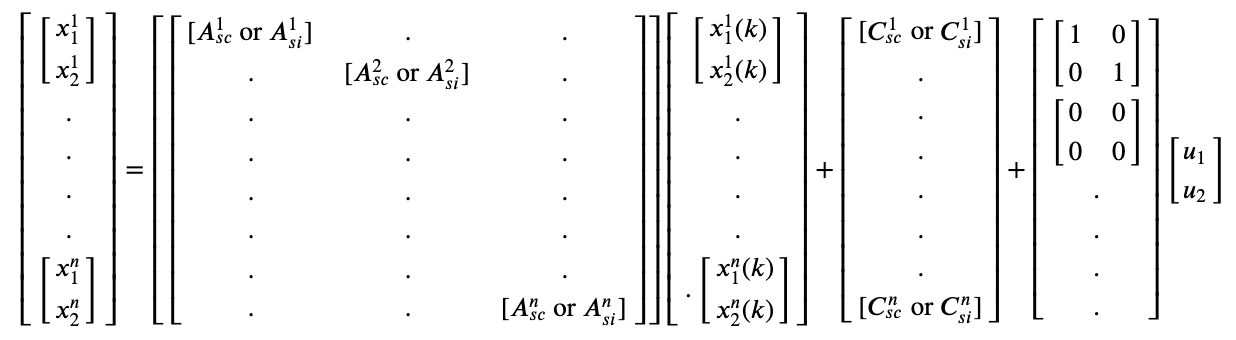
## 3.5 Considerations for Vessels

Vessels that go to the buoys to stock up on water and energy must be integrated into the model. You can think of adding variables that represent the quantity of resources taken from the boats:

* **u1**: Energy withdrawn.
* **u2**: Water taken.

We could hypothesize the use of a Δt equal to one, so as to consider all charging and discharging phenomena as punctual and instantaneous.

The full model could then include these withdrawals in the dynamic equations:



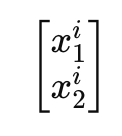
summarizing in detail:

### 3.5.1 State of the Buoys

Each buoy has two state variables:

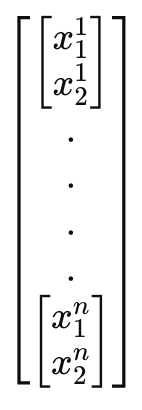
* x1i: Energy accumulated in buoy i at time k.
* x2i: Water accumulated in buoy i at time k.

These variables are collected in a column vector for each buoy:



### 3.5.1 Complete System of Buoys

The complete system representation with n buoys is:

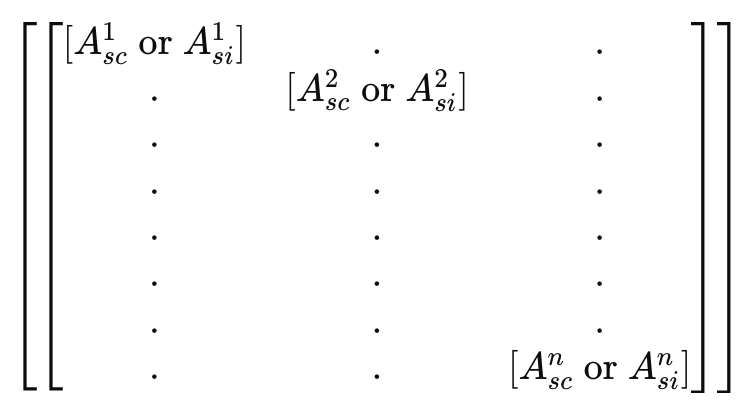


### 3.5.2 Transition Matrix

The system dynamics are described by a transition matrix that can vary depending on whether there is a complete or incomplete discharge of the accumulated energy. This matrix is composed of sub-matrices **Asci​** or ​ **Asii**, where:

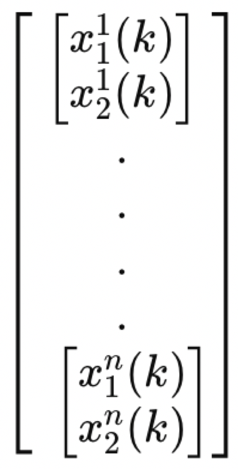
* ​**Asc**: Represents the case of complete discharge.
* **Asi**: Represents the case of incomplete discharge.

The complete transition matrix is:



### 3.5.3 State Vector at Time k

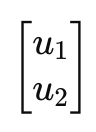
The state vector of the buoys at time k is:



where the superscript indicates the buoy number and the subscript indicates the x1 or x2 status of the single buoy.

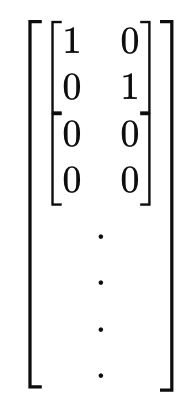
### 3.5.4 Input Vector

The input vector represents the external contributions of energy and water required by the buoys:



### 3.5.5 Matrix Input Vector

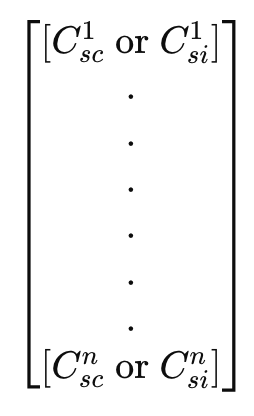
The matrix that multiplies the input vector takes into account the resources withdrawn:



specifically, the use of these matrix subblocks indicates the assignment of the vessel to a buoy, the request will therefore arrive at the buoy identified by a certain index i whose matrix subblock will have the 1's positioned on the diagonal.

### 3.5.6 Vector of Constant Terms

The vector of constant terms represents the fixed contributions of energy and water:



In summary, this matrix form represents the dynamic evolution of a system of buoys that accumulate energy and water, considering the internal dynamics of each buoy and the external contributions in terms of resources taken.

# 4. Code implementation and results

In implementing the code we opted for different scenarios, listed below:

1. Implementation of a single autonomous system consisting of 3 buoys with random generation of a number of vessels (for example a maximum of 2) per instant of time. The assignment occurs simply to the first buoy that has the resources available to satisfy the offer.
2. Implementation of three autonomous systems each consisting of 3 buoys with random generation of vessel requests for time slots, specifically:
   1. 02:00am – 05:00am: zero installments (0)
   2. 06:00am - 10:00am: low rate (1 to max\_vessels / 3)
   3. 10:00am -18:00pm: high rate (1 to max\_vessels)
   4. 18:00pm - 22:00pm: low rate (1 to max\_vessels / 3)
   5. 10:00pm - 06:00am: rate almost zero (0 to 1)

the assignment takes place at the first mark with available resources.

1. Implementation of a single autonomous system consisting of 3 buoys with random generation of a number of vessels (for example a maximum of 2) per instant of time. The assignment takes place according to a round robin criterion.
2. Implementation of three autonomous systems each consisting of 3 buoys with random generation of vessel requests for time slots, specifically:
   1. 02:00am – 05:00am: zero installments (0)
   2. 06:00am - 10:00am: low rate (1 to max\_vessels / 3)
   3. 10:00am -18:00pm: high rate (1 to max\_vessels)
   4. 18:00pm - 22:00pm: low rate (1 to max\_vessels / 3)
   5. 10:00pm - 06:00am: rate almost zero (0 to 1)

the assignment occurs according to a round robin criterion even between autonomous systems.

1. Implementation of three autonomous systems each consisting of 3 buoys with random generation of vessel requests for time slots, specifically:
   1. 02:00am – 05:00am: zero installments (0)
   2. 06:00am - 10:00am: low rate (1 to max\_vessels / 3)
   3. 10:00am -18:00pm: high rate (1 to max\_vessels)
   4. 18:00pm - 22:00pm: low rate (1 to max\_vessels / 3)
   5. 10:00pm - 06:00am: rate almost zero (0 to 1)

The assignment takes place according to the following method:

* Data Collection: Each buoy collects its own status data (available energy and water).
* Data Sharing: Buoys share their status data with other buoys in the system.
* Data Verification: Each buoy verifies the data received from the other buoys by comparing them with its own data and with the data received from other buoys.
* Data Consensus: Buoys reach a consensus on the correct data through a consensus protocol such as the Byzantine Generals Protocol.

This mode manages the possibility that some buoys may provide incorrect values ​​to every other buoy thanks to a completely distributed communication throughout the entire network of buoys, and also makes resource consumption efficient by maximizing it on each individual buoy.

1. It is an evolved version of problem 5 which exploits the problem of the Byzantine generals for the exchange of information between different autonomous systems, in this mode there will be a commander for each autonomous system who will then direct within his own autonomous system or will redirect to another.

For brevity we will only analyze complete cases with multiple ASs, specifically 2, 4, 5 and 6.

## 4.1 Case n°2

*Global Parameters*

The initial parameters are defined, including:

* The number of buoy systems (num\_autonomous\_systems = 3).
* The number of buoys per system (n = 3).
* The constant energy production (cr = 4.6 units per timestep).
* The energy-water conversion parameter (kc = 0.05).
* The simulation duration (timesteps = 24).
* Initial levels of energy and water (initial\_energy = 5 and initial\_water = 10).
* Maximum levels of energy and water (max\_energy = 5 and max\_water = 10).
* Minimum energy threshold (min\_energy = 1).
* Target water level (x2\_target = 10).
* Maximum number of vessels per timestep (max\_vessels = 10).

*State Vector Initialization*

The initialize\_state\_vectors function creates and initializes the state vectors for energy and water for each buoy with the updated initial conditions.

*Vessel Number Generation*

The generate\_num\_vessels\_per\_timestep function generates the number of vessels for each timestep, following a random distribution with varying rates of vessel arrivals.

*Random Request Generation*

The generate\_random\_requests function creates random requests for energy and water from the vessels arriving at each timestep.

*Buoy State Update*

The update\_buoys function updates the energy and water levels of the buoys based on production and energy-water conversion. It ensures that the levels remain within the defined maximum thresholds.

*Request Assignment*

The assign\_requests\_to\_systems function assigns vessel requests to the buoys, updating their energy and water levels to balance the load across all buoys.

*Simulation Execution*

The run\_simulation\_multiple\_systems function runs the simulation for the specified number of timesteps, updating buoy states and assigning vessel requests at each timestep.

*Results Visualization*

The plot\_results\_multiple\_systems function visualizes the simulation results, displaying the energy and water levels for each buoy over time. It includes labels for each timestep, formatted as hours of the day.

***Summary of Results***

*Energy Levels*

* The energy levels of all buoys in each system show significant fluctuations throughout the simulation.
* The energy levels start at the maximum threshold (5 units) and fluctuate due to the varying demands from the vessels.
* The continuous rise and fall in energy levels suggest a balance between energy production and consumption by the vessels.

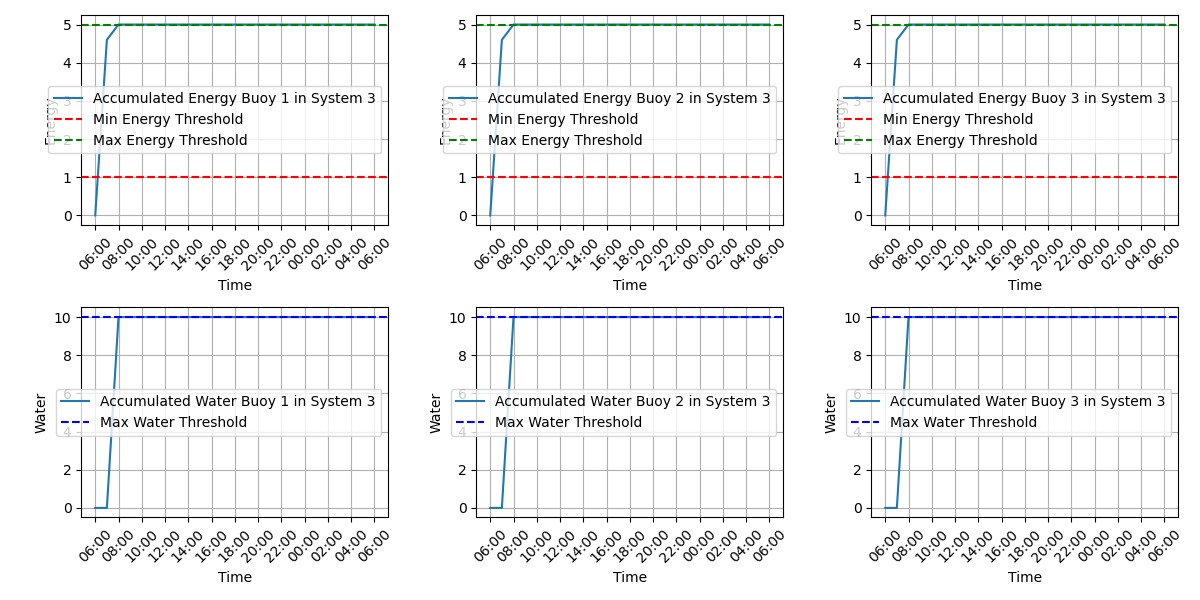
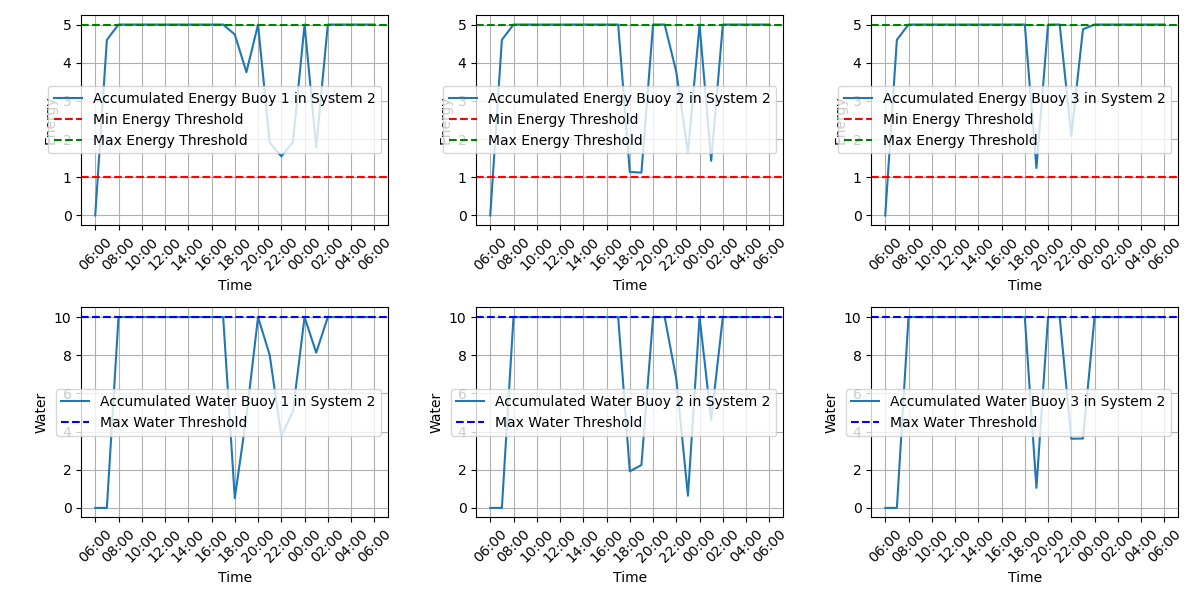
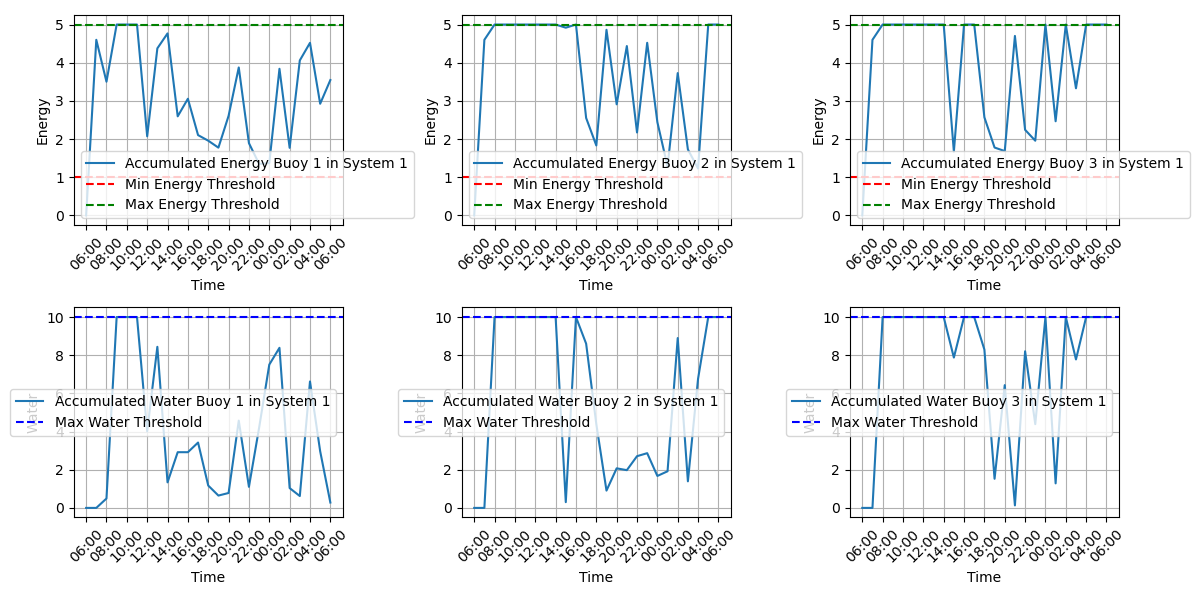
*Water Levels*

* The water levels exhibit considerable variations, starting at the maximum threshold (10 units).
* There are instances where the water levels drop to lower values, indicating high water usage by the vessels and efficient replenishment cycles.

*Request Assignment*

* The assignment method ensures balanced utilization of all buoys, preventing any single buoy from being overloaded.
* The method effectively assigns requests while maintaining energy and water levels within defined thresholds, ensuring reliable operation.

### 4.1.1 Results



The provided graphs display the accumulated energy and water levels for each buoy in three autonomous buoy systems over a 24-hour simulation period. Below is an analysis of the results for each system.

System 1

**Energy Levels:**

* **Buoy 1**: The energy levels fluctuate significantly throughout the simulation. The energy often reaches the maximum threshold of 5 units and occasionally approaches the minimum threshold of 1 unit, indicating high energy demand and usage.
* **Buoy 2**: The energy levels start at the maximum and frequently drop close to the minimum threshold, showing similar usage patterns as Buoy 1.
* **Buoy 3**: The energy levels also exhibit significant fluctuations, reaching the maximum threshold multiple times, suggesting effective energy management and high energy consumption.

**Water Levels:**

* **Buoy 1**: The water levels fluctuate, often nearing the maximum threshold of 10 units but also showing significant drops, indicating frequent water usage and replenishment.
* **Buoy 2**: The water levels exhibit frequent fluctuations, similar to Buoy 1, suggesting consistent water usage by vessels.
* **Buoy 3**: The water levels show regular variations, often dropping to lower values, indicating efficient water management and high demand.

System 2

**Energy Levels:**

* **Buoy 1**: The energy levels start at the maximum threshold and show significant fluctuations, similar to System 1, indicating high energy demand and usage.
* **Buoy 2**: The energy levels frequently reach the maximum threshold and drop close to the minimum threshold, showing high usage.
* **Buoy 3**: The energy levels also exhibit significant fluctuations, similar to the other buoys, indicating balanced energy production and consumption.

**Water Levels:**

* **Buoy 1**: The water levels show considerable variation, often reaching the maximum threshold and dropping to lower values, indicating frequent usage.
* **Buoy 2**: The water levels exhibit frequent drops and replenishment cycles, similar to Buoy 1.
* **Buoy 3**: The water levels show regular variations, indicating consistent water usage and replenishment.

System 3

**Energy Levels:**

* **Buoy 1**: The energy levels quickly reach the maximum threshold and remain there, indicating no significant energy usage.
* **Buoy 2**: Similar to Buoy 1, the energy levels quickly reach and stay at the maximum threshold, indicating a lack of energy demand.
* **Buoy 3**: The energy levels also quickly reach the maximum threshold and remain constant, suggesting no significant energy consumption.

**Water Levels:**

* **Buoy 1**: The water levels quickly reach the maximum threshold and stay there, indicating no significant water usage.
* **Buoy 2**: Similar to Buoy 1, the water levels quickly reach and stay at the maximum threshold.
* **Buoy 3**: The water levels also quickly reach and remain at the maximum threshold, suggesting no significant water demand.

### 4.1.2 General Observations

1. **Fluctuation Patterns**: Systems 1 and 2 show similar patterns of significant fluctuations in energy and water levels, indicating high demand and usage by vessels. System 3, however, shows little to no fluctuation, suggesting no significant demand for energy or water.
2. **Threshold Compliance**: The energy and water levels generally remain within the defined thresholds, ensuring that the buoys do not exceed their storage capacities.
3. **Demand-Driven Consumption**: The frequent drops in both energy and water levels in Systems 1 and 2 suggest a high and regular demand from the vessels, requiring the buoys to continuously supply resources. System 3's constant levels suggest a lack of demand.

These results reflect the efficiency and responsiveness of the autonomous buoy systems in managing energy and water resources under varying demand conditions, with Systems 1 and 2 experiencing high usage and System 3 experiencing minimal usage.

## 4.2 Case n°4

*Global Parameters*

The initial parameters are defined, including:

* The number of buoy systems (num\_autonomous\_systems = 3).
* The number of buoys per system (n = 3).
* The constant energy production (cr = 4.6 units per timestep).
* The energy-water conversion parameter (kc = 0.05).
* The simulation duration (timesteps = 24).
* Initial levels of energy and water (initial\_energy = 0 and initial\_water = 0).
* Maximum levels of energy and water (max\_energy = 5 and max\_water = 10).
* Minimum energy threshold (min\_energy = 1).
* Target water level (x2\_target = 10).
* Maximum number of vessels per timestep (max\_vessels = 10).

*State Vector Initialization*

The initialize\_state\_vectors function creates and initializes the state vectors for energy and water for each buoy.

*Vessel Number Generation*

The generate\_num\_vessels\_per\_timestep function generates the number of vessels for each timestep, following a random distribution with varying rates of vessel arrivals.

*Random Request Generation*

The generate\_random\_requests function creates random requests for energy and water from the vessels arriving at each timestep.

*Buoy State Update*

The update\_buoys function updates the energy and water levels of the buoys based on production and energy-water conversion. It ensures that the levels remain within the defined maximum thresholds.

*Request Assignment*

The assign\_requests\_round\_robin function assigns vessel requests to the buoys using a round-robin method, balancing the load across all buoys and ensuring that energy and water levels are updated correctly.

*Simulation Execution*

The run\_simulation\_round\_robin function runs the simulation for the specified number of timesteps, updating buoy states and assigning vessel requests at each timestep using the round-robin method.

*Results Visualization*

The plot\_results\_multiple\_systems function visualizes the simulation results, displaying the energy and water levels for each buoy over time.

**Summary of Results**

*Energy Levels*

* The energy levels of all buoys in each system show significant fluctuations throughout the simulation.
* The energy levels often reach the maximum threshold (5 units) and occasionally approach the minimum threshold (1 unit), indicating efficient energy management and high demand.
* The continuous rise and fall in energy levels suggest a balance between energy production and consumption by the vessels.

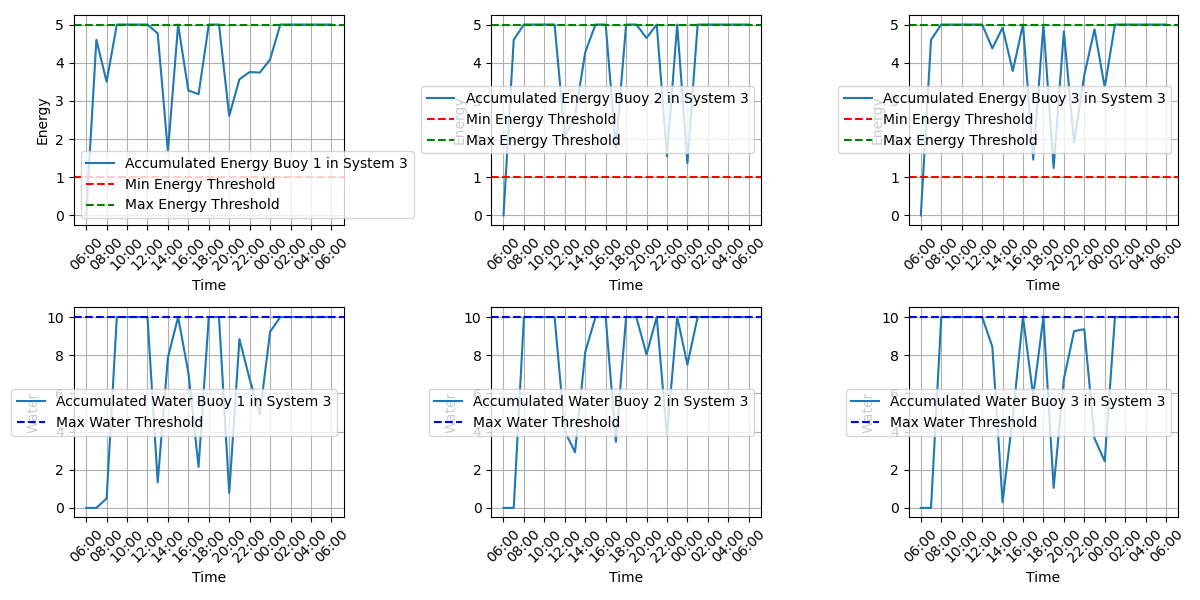
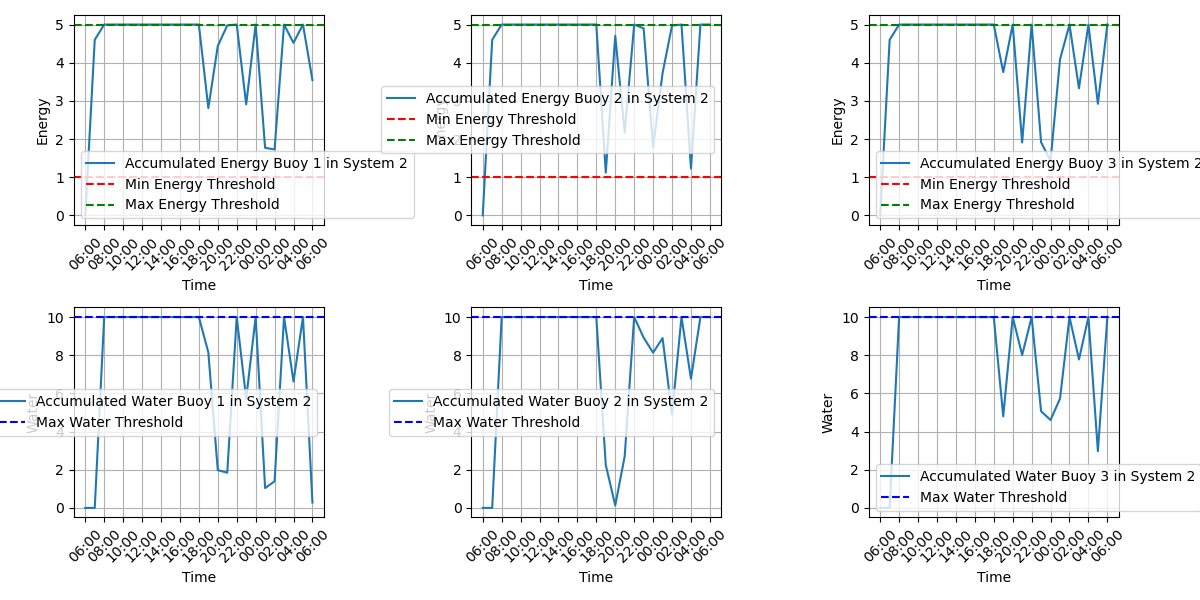
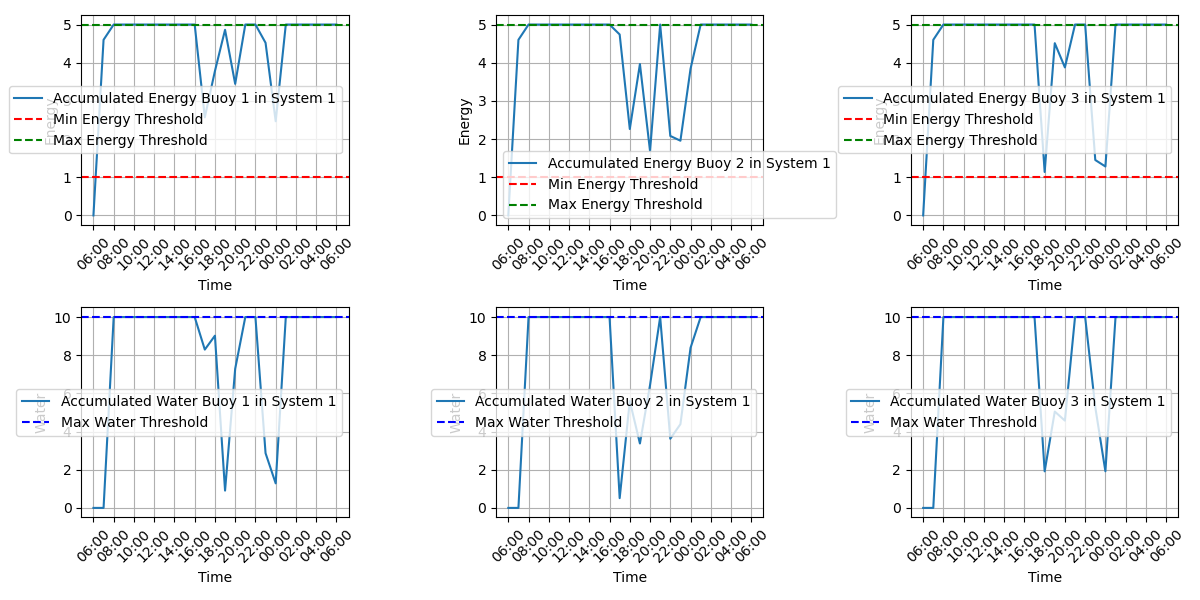
*Water Levels*

* The water levels exhibit considerable variations, frequently reaching the maximum threshold (10 units).
* There are multiple instances where the water levels drop to zero, indicating high water usage by the vessels and efficient replenishment cycles.

*Round Robin Assignment*

* The round-robin method ensures balanced utilization of all buoys, preventing any single buoy from being overloaded.
* The method effectively assigns requests while maintaining energy and water levels within defined thresholds, ensuring reliable operation.

### 4.2.1 Results



The provided graphs display the accumulated energy and water levels for each buoy in three autonomous buoy systems over a 24-hour simulation period. Below is an analysis of the results for each system.

System 1

**Energy Levels:**

* **Buoy 1**: The energy levels fluctuate significantly throughout the simulation. The energy often reaches the maximum threshold of 5 units and occasionally approaches the minimum threshold of 1 unit, indicating high energy demand and usage.
* **Buoy 2**: The energy levels start at the maximum and frequently drop close to the minimum threshold, showing similar usage patterns as Buoy 1.
* **Buoy 3**: The energy levels also exhibit significant fluctuations, reaching the maximum threshold multiple times, suggesting effective energy management and high energy consumption.

**Water Levels:**

* **Buoy 1**: The water levels fluctuate, often nearing the maximum threshold of 10 units but also showing significant drops, indicating frequent water usage and replenishment.
* **Buoy 2**: The water levels exhibit frequent fluctuations, similar to Buoy 1, suggesting consistent water usage by vessels.
* **Buoy 3**: The water levels show regular variations, often dropping to lower values, indicating efficient water management and high demand.

System 2

**Energy Levels:**

* **Buoy 1**: The energy levels start at the maximum threshold and show significant fluctuations, similar to System 1, indicating high energy demand and usage.
* **Buoy 2**: The energy levels frequently reach the maximum threshold and drop close to the minimum threshold, showing high usage.
* **Buoy 3**: The energy levels also exhibit significant fluctuations, similar to the other buoys, indicating balanced energy production and consumption.

**Water Levels:**

* **Buoy 1**: The water levels show considerable variation, often reaching the maximum threshold and dropping to lower values, indicating frequent usage.
* **Buoy 2**: The water levels exhibit frequent drops and replenishment cycles, similar to Buoy 1.
* **Buoy 3**: The water levels show regular variations, indicating consistent water usage and replenishment.

System 3

**Energy Levels:**

* **Buoy 1**: The energy levels start at a lower level and gradually reach the maximum threshold. Fluctuations are less pronounced compared to Systems 1 and 2.
* **Buoy 2**: The energy levels follow a similar pattern, gradually increasing and showing moderate fluctuations.
* **Buoy 3**: The energy levels also gradually increase and exhibit moderate fluctuations, suggesting a balanced energy usage pattern.

**Water Levels:**

* **Buoy 1**: The water levels quickly reach the maximum threshold and show less fluctuation, indicating a period of less water demand.
* **Buoy 2**: The water levels exhibit moderate fluctuations, indicating some periods of high water usage followed by replenishment.
* **Buoy 3**: The water levels follow a similar pattern, showing moderate fluctuations and periods of replenishment.

### 4.2.2 General Observations

1. **Fluctuation Patterns**: Systems 1 and 2 show significant fluctuations in energy and water levels, indicating high demand and usage by vessels. System 3 shows moderate fluctuations, suggesting a period of less demand.
2. **Threshold Compliance**: The energy and water levels generally remain within the defined thresholds, ensuring that the buoys do not exceed their storage capacities.
3. **Demand-Driven Consumption**: The frequent drops in both energy and water levels in Systems 1 and 2 suggest high and regular demand from the vessels, requiring the buoys to continuously supply resources. System 3's moderate fluctuations suggest a balanced usage pattern.

These results reflect the efficiency and responsiveness of the autonomous buoy systems in managing energy and water resources under varying demand conditions, with Systems 1 and 2 experiencing high usage and System 3 experiencing moderate usage.

## 4.3 Case n°6

*Global Parameters*

The initial parameters are defined, including:

* The number of buoy systems (num\_autonomous\_systems = 3).
* The number of buoys per system (n = 3).
* The constant energy production (cr = 4.6 units per timestep).
* The energy-water conversion parameter (kc = 0.05).
* The simulation duration (timesteps = 24).
* Initial levels of energy and water (initial\_energy = 5 and initial\_water = 10).
* Maximum levels of energy and water (max\_energy = 5 and max\_water = 10).
* Minimum energy threshold (min\_energy = 1).
* Target water level (x2\_target = 10).
* Maximum number of vessels per timestep (max\_vessels = 9).

*State Vector Initialization*

The initialize\_state\_vectors function creates and initializes the state vectors for energy and water for each buoy with the updated initial conditions.

*Vessel Number Generation*

The generate\_num\_vessels\_per\_timestep function generates the number of vessels for each timestep, following a random distribution with varying rates of vessel arrivals.

*Random Request Generation*

The generate\_random\_requests function creates random requests for energy and water from the vessels arriving at each timestep.

*Buoy State Update*

The update\_buoys function updates the energy and water levels of the buoys based on production and energy-water conversion. It ensures that the levels remain within the defined maximum thresholds.

*Request Assignment*

The assign\_requests\_optimized function assigns vessel requests to the buoys using an optimized greedy algorithm, which minimizes the impact on the buoy's resources.

*Data Verification and Consensus*

The verify\_and\_consensus function ensures that the data collected by each buoy is accurate by using a consensus mechanism, where the most common value among reported values is chosen as the verified data.

*Simulation Execution*

The run\_simulation\_optimized function runs the simulation for the specified number of timesteps, updating buoy states, verifying data, and assigning vessel requests at each timestep using the optimized greedy algorithm.

*Results Visualization*

The plot\_results\_multiple\_systems function visualizes the simulation results, displaying the energy and water levels for each buoy over time.

**Summary of Results**

*Energy Levels*

* The energy levels of all buoys in each system show significant fluctuations throughout the simulation.
* The energy levels start at the maximum threshold (5 units) and fluctuate due to the varying demands from the vessels.
* The continuous rise and fall in energy levels suggest a balance between energy production and consumption by the vessels.

*Water Levels*

* The water levels exhibit considerable variations, starting at the maximum threshold (10 units).
* There are instances where the water levels drop to lower values, indicating high water usage by the vessels and efficient replenishment cycles.

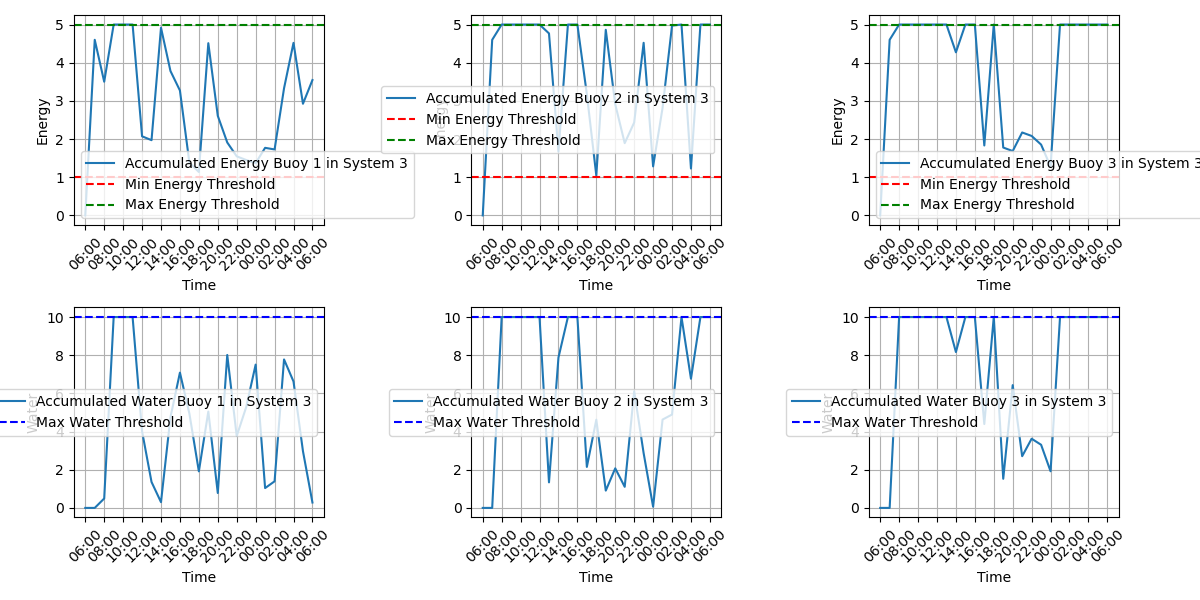
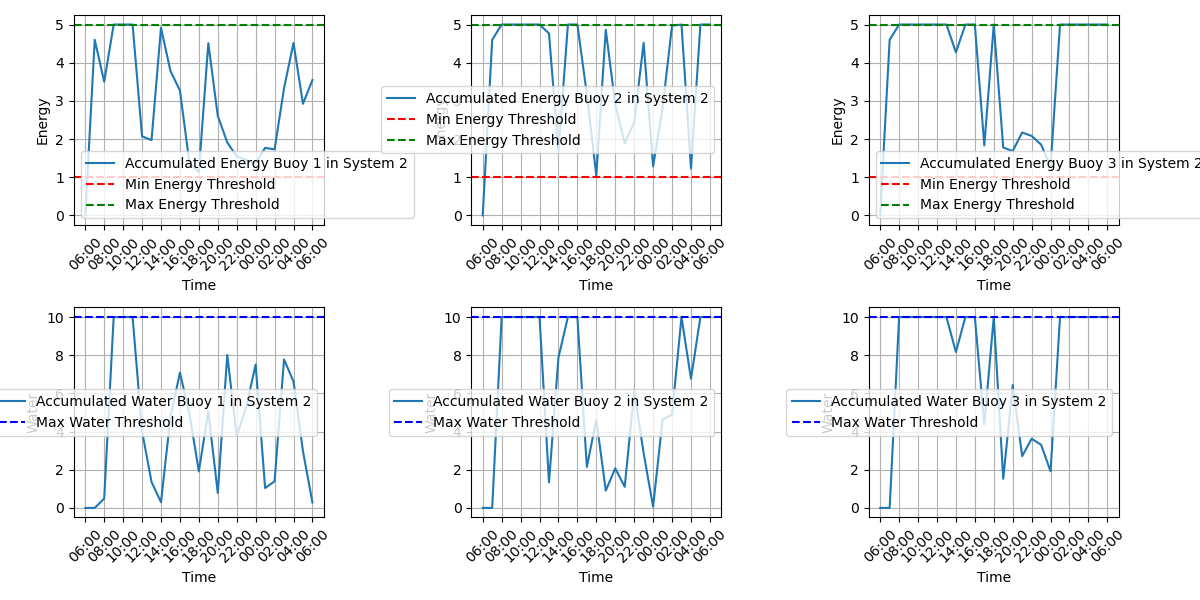
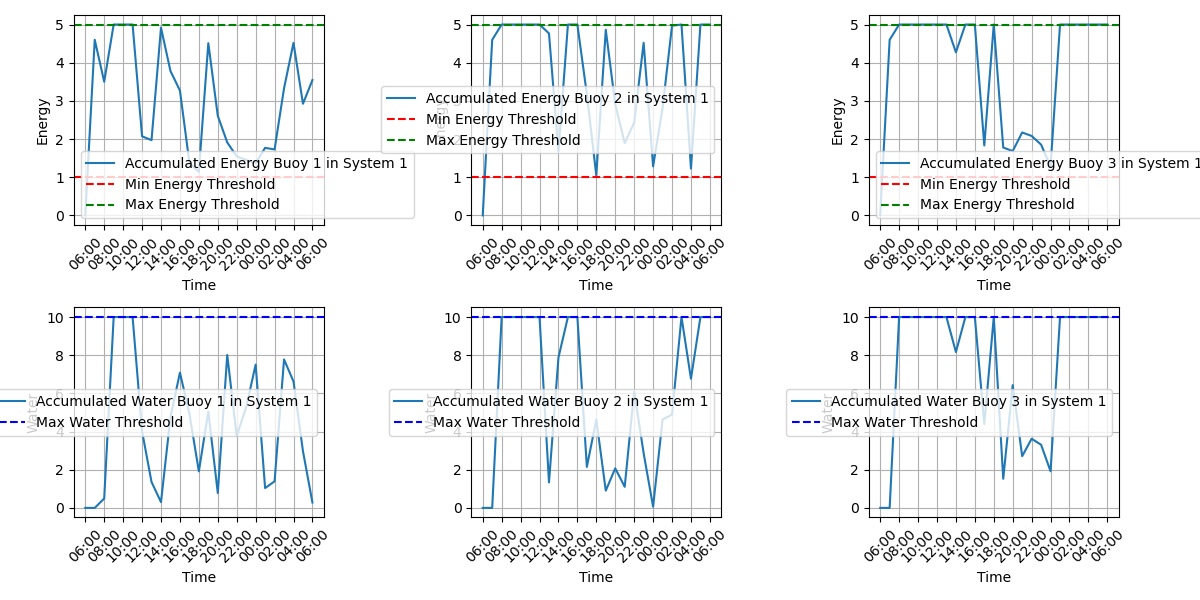
*Optimized Greedy Assignment*

* The optimized greedy algorithm ensures efficient utilization of buoy resources by minimizing the impact of each request.
* The method effectively assigns requests while maintaining energy and water levels within defined thresholds, ensuring reliable operation.

*Data Verification and Consensus*

* The consensus mechanism ensures data accuracy by verifying the reported values among the buoys and updating the state vectors accordingly.
* This process helps in maintaining the integrity of the simulation data and provides robust results.

### 4.3.1 Results



The provided graphs display the accumulated energy and water levels for each buoy in three autonomous buoy systems over a 24-hour simulation period. Below is an analysis of the results for each system.

System 1

**Energy Levels:**

* **Buoy 1**: The energy levels fluctuate significantly, often nearing the maximum threshold of 5 units and occasionally dropping close to the minimum threshold of 1 unit. This suggests high energy demand and effective usage.
* **Buoy 2**: Similar to Buoy 1, the energy levels fluctuate considerably, indicating frequent energy consumption and replenishment.
* **Buoy 3**: The energy levels show notable fluctuations, suggesting consistent energy demand and effective resource management.

**Water Levels:**

* **Buoy 1**: The water levels fluctuate, with the buoy often approaching the maximum water threshold of 10 units and experiencing significant drops, indicating active water usage.
* **Buoy 2**: The water levels exhibit similar patterns to Buoy 1, showing consistent fluctuations and indicating regular water demand.
* **Buoy 3**: The water levels also fluctuate, reflecting regular usage and efficient water management.

System 2

**Energy Levels:**

* **Buoy 1**: The energy levels start at the maximum threshold and exhibit significant fluctuations, similar to System 1, indicating high energy usage.
* **Buoy 2**: The energy levels frequently reach the maximum threshold and drop close to the minimum threshold, showing substantial usage.
* **Buoy 3**: The energy levels also show significant fluctuations, reflecting balanced energy production and consumption.

**Water Levels:**

* **Buoy 1**: The water levels show considerable variation, often reaching the maximum threshold and experiencing drops, indicating frequent usage.
* **Buoy 2**: The water levels exhibit frequent drops and replenishment cycles, similar to Buoy 1.
* **Buoy 3**: The water levels show regular variations, indicating consistent water usage and replenishment.

System 3

**Energy Levels:**

* **Buoy 1**: The energy levels start at a lower level and quickly reach the maximum threshold. Fluctuations are less pronounced compared to Systems 1 and 2, suggesting lower demand.
* **Buoy 2**: The energy levels follow a similar pattern, quickly reaching and showing moderate fluctuations.
* **Buoy 3**: The energy levels also quickly increase and exhibit moderate fluctuations, indicating balanced energy usage.

**Water Levels:**

* **Buoy 1**: The water levels quickly reach the maximum threshold and show less fluctuation, indicating a period of less water demand.
* **Buoy 2**: The water levels exhibit moderate fluctuations, indicating periods of high water usage followed by replenishment.
* **Buoy 3**: The water levels follow a similar pattern, showing moderate fluctuations and periods of replenishment.

### 4.3.2 General Observations

1. **Fluctuation Patterns**: Systems 1 and 2 show significant fluctuations in energy and water levels, indicating high demand and usage by vessels. System 3 shows moderate fluctuations, suggesting a period of less demand.
2. **Threshold Compliance**: The energy and water levels generally remain within the defined thresholds, ensuring that the buoys do not exceed their storage capacities.
3. **Demand-Driven Consumption**: The frequent drops in both energy and water levels in Systems 1 and 2 suggest high and regular demand from the vessels, requiring the buoys to continuously supply resources. System 3's moderate fluctuations suggest a balanced usage pattern.

These results reflect the efficiency and responsiveness of the autonomous buoy systems in managing energy and water resources under varying demand conditions, with Systems 1 and 2 experiencing high usage and System 3 experiencing moderate usage.

## 4.4 Case n°7

*Global Parameters*

The initial parameters are defined, including:

* The number of buoy systems (num\_autonomous\_systems = 3).
* The number of buoys per system (n = 3).
* The constant energy production (cr = 4.6 units per timestep).
* The energy-water conversion parameter (kc = 0.05).
* The simulation duration (timesteps = 24).
* Initial levels of energy and water (initial\_energy = 1 and initial\_water = 0).
* Maximum levels of energy and water (max\_energy = 5 and max\_water = 10).
* Minimum energy threshold (min\_energy = 1).
* Target water level (x2\_target = 10).
* Maximum number of vessels per timestep (max\_vessels = 9).

*State Vector Initialization*

The initialize\_state\_vectors function creates and initializes the state vectors for energy and water for each buoy with the specified initial conditions.

*Vessel Number Generation*

The generate\_num\_vessels\_per\_timestep function generates the number of vessels for each timestep, following a random distribution with varying rates of vessel arrivals.

*Random Request Generation*

The generate\_random\_requests function creates random requests for energy and water from the vessels arriving at each timestep.

*Buoy State Update*

The update\_buoys function updates the energy and water levels of the buoys based on production and energy-water conversion. It ensures that the levels remain within the defined maximum thresholds.

*Request Assignment*

The assign\_requests\_to\_systems function assigns vessel requests to the buoys within each system. It uses the assign\_requests\_to\_buoys function to optimize the assignment by minimizing the impact on buoy resources.

*Data Verification and Consensus*

The verify\_and\_consensus function ensures that the data collected by each buoy is accurate by using a consensus mechanism, where the most common value among reported values is chosen as the verified data.

*Simulation Execution*

The run\_simulation\_optimized function runs the simulation for the specified number of timesteps, updating buoy states, verifying data, and assigning vessel requests at each timestep using the optimized assignment algorithm.

*Results Visualization*

The plot\_results\_multiple\_systems function visualizes the simulation results, displaying the energy and water levels for each buoy over time. It includes labels for each timestep, formatted as hours of the day.

**Summary of Results**

*Energy Levels*

* The energy levels of all buoys in each system show significant fluctuations throughout the simulation.
* The energy levels start at a lower initial level (1 unit) and fluctuate due to the varying demands from the vessels.
* The continuous rise and fall in energy levels suggest a balance between energy production and consumption by the vessels.

*Water Levels*

* The water levels exhibit considerable variations, starting at the initial level (0 units).
* There are instances where the water levels increase due to the conversion of energy to water, indicating efficient replenishment cycles.

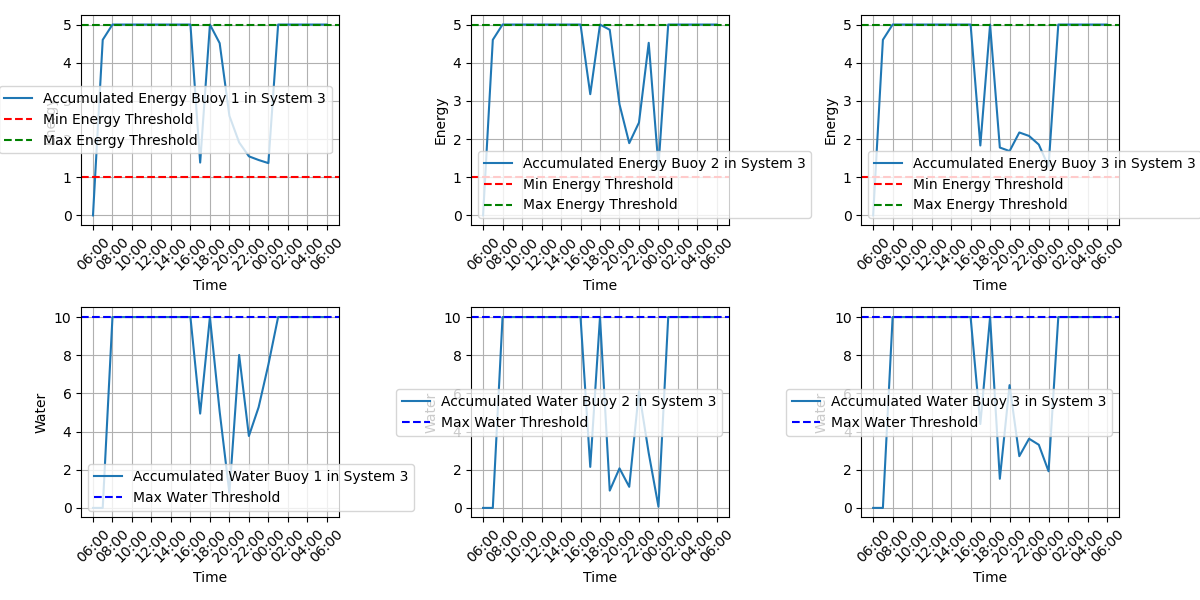
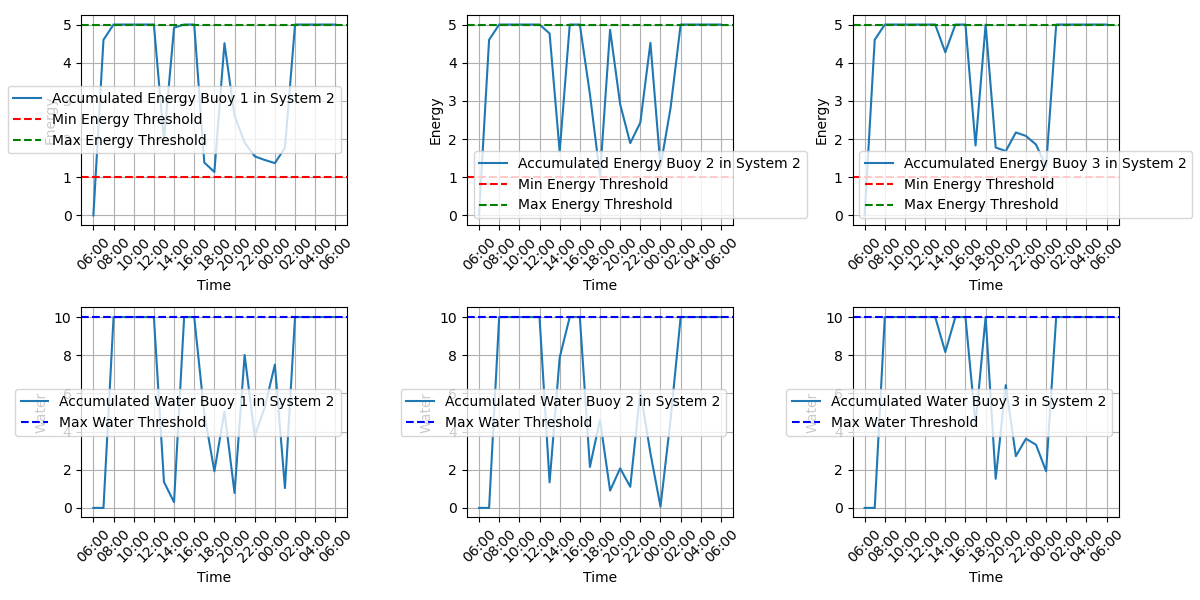
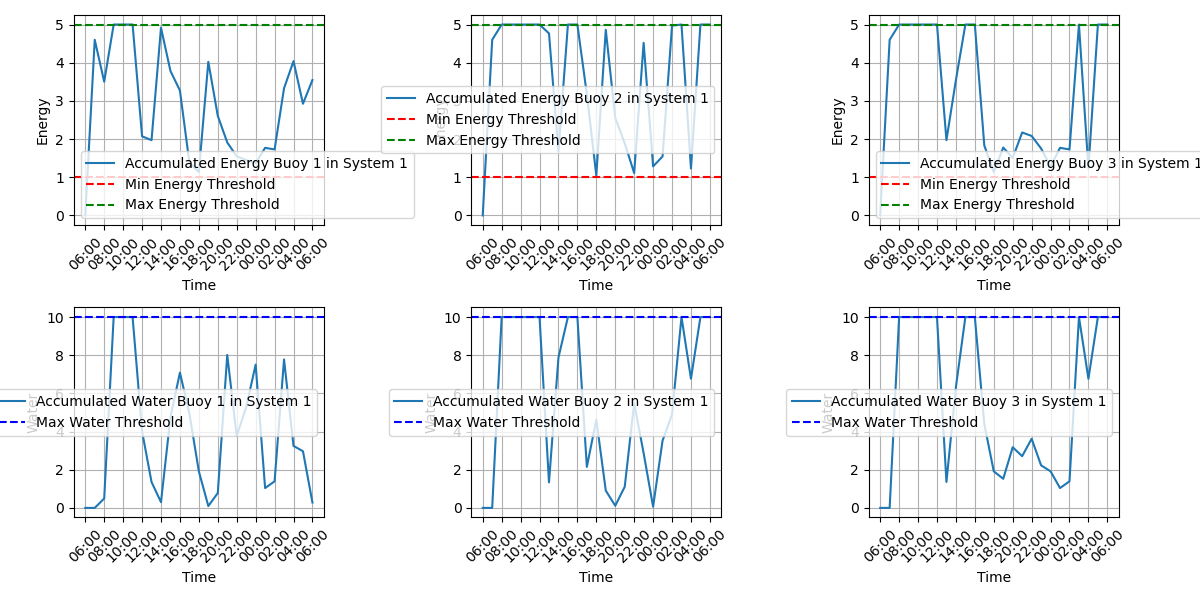
*Optimized Assignment*

* The optimized assignment algorithm ensures efficient utilization of buoy resources by minimizing the impact of each request.
* The method effectively assigns requests while maintaining energy and water levels within defined thresholds, ensuring reliable operation.

*Data Verification and Consensus*

* The consensus mechanism ensures data accuracy by verifying the reported values among the buoys and updating the state vectors accordingly.
* This process helps in maintaining the integrity of the simulation data and provides robust results.

### 4.4.1 Results



The provided graphs display the accumulated energy and water levels for each buoy in three autonomous buoy systems over a 24-hour simulation period. Below is an analysis of the results for each system.

System 1

**Energy Levels:**

* **Buoy 1:** The energy levels fluctuate significantly, often nearing the maximum threshold of 5 units and occasionally dropping close to the minimum threshold of 1 unit. This suggests high energy demand and effective usage.
* **Buoy 2:** Similar to Buoy 1, the energy levels fluctuate considerably, indicating frequent energy consumption and replenishment.
* **Buoy 3:** The energy levels show notable fluctuations, suggesting consistent energy demand and effective resource management.

**Water Levels:**

* **Buoy 1:** The water levels fluctuate, with the buoy often approaching the maximum water threshold of 10 units and experiencing significant drops, indicating active water usage.
* **Buoy 2:** The water levels exhibit similar patterns to Buoy 1, showing consistent fluctuations and indicating regular water demand.
* **Buoy 3:** The water levels also fluctuate, reflecting regular usage and efficient water management.

System 2

**Energy Levels:**

* **Buoy 1:** The energy levels start at the maximum threshold and exhibit significant fluctuations, similar to System 1, indicating high energy usage.
* **Buoy 2:** The energy levels frequently reach the maximum threshold and drop close to the minimum threshold, showing substantial usage.
* **Buoy 3:** The energy levels also show significant fluctuations, reflecting balanced energy production and consumption.

**Water Levels:**

* **Buoy 1:** The water levels show considerable variation, often reaching the maximum threshold and experiencing drops, indicating frequent usage.
* **Buoy 2:** The water levels exhibit frequent drops and replenishment cycles, similar to Buoy 1.
* **Buoy 3:** The water levels show regular variations, indicating consistent water usage and replenishment.

System 3

**Energy Levels:**

* **Buoy 1:** The energy levels start at a lower level and quickly reach the maximum threshold. Fluctuations are less pronounced compared to Systems 1 and 2, suggesting lower demand.
* **Buoy 2:** The energy levels follow a similar pattern, quickly reaching and showing moderate fluctuations.
* **Buoy 3:** The energy levels also quickly increase and exhibit moderate fluctuations, indicating balanced energy usage.

**Water Levels:**

* **Buoy 1:** The water levels quickly reach the maximum threshold and show less fluctuation, indicating a period of less water demand.
* **Buoy 2:** The water levels exhibit moderate fluctuations, indicating periods of high water usage followed by replenishment.
* **Buoy 3:** The water levels follow a similar pattern, showing moderate fluctuations and periods of replenishment.

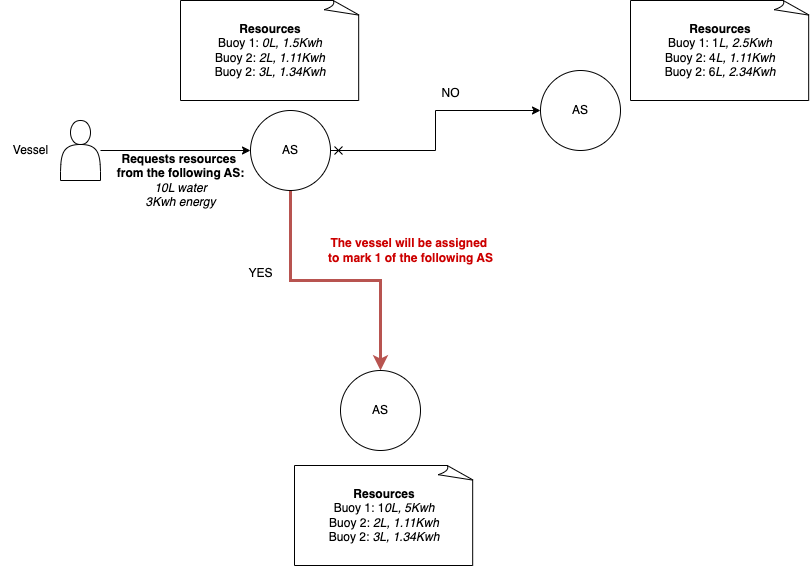
### 4.4.2 General Observations

* **Fluctuation Patterns:** Systems 1 and 2 show significant fluctuations in energy and water levels, indicating high demand and usage by vessels. System 3 shows moderate fluctuations, suggesting a period of less demand.
* **Threshold Compliance:** The energy and water levels generally remain within the defined thresholds, ensuring that the buoys do not exceed their storage capacities.
* **Demand-Driven Consumption:** The frequent drops in both energy and water levels in Systems 1 and 2 suggest high and regular demand from the vessels, requiring the buoys to continuously supply resources. System 3's moderate fluctuations suggest a balanced usage pattern.

These results reflect the efficiency and responsiveness of the autonomous buoy systems in managing energy and water resources under varying demand conditions, with Systems 1 and 2 experiencing high usage and System 3 experiencing moderate usage.

# 5 Request the distribution model between and within the ASes and assign requests

In the examples above we have addressed different request distribution patterns, each with advantages and disadvantages. The objective is to assign a request for a vessel to a buoy: all the cases that will be described will have the same condition in common, in which, if none of the buoys possess the requested resources, it will send the vessel to another AS. Below is an example that describes its behavior:



What changes in the following models will be related to the maximization of resource consumption at the buoy and the robustness of the data exchange which could be subject to sensor errors.

We therefore analyze the different cases and describe how they work.

## 5.1 Simple method

In this scenario we have a distribution done in the following way: the first buoy that can provide the resources is the one that will satisfy the request.

## 5.2 Round Robin method

Immagine che contiene diagramma, schermata, linea, testo

Descrizione generata automaticamente

In an iterative manner, the vessel is assigned in rotation to each buoy of the AS. If there is no buoy in the AS that can satisfy this request, it will send the vessel back to the first AS capable of satisfying it.

*Description of the method in the context of origin*

The Round Robin method is a scheduling algorithm used in operating systems to manage the execution of processes in a fair and balanced way. It is especially popular in time-sharing systems, where the goal is to provide a quick interactive experience to all users. Here is a detailed explanation of how it works:

Basic Principle

The requests will be taken care of and distributed in rotation among the buoys relating to a certain AS. Various cases may occur:

* a certain buoy of the AS cannot satisfy the request, in this case the request is forwarded to the next one which will perform the same check;
* no buoy within the AS can satisfy the request, in which case the vessel will be redirected to another AS which will have at least one buoy that can satisfy the request.

on each AS the load distribution on the buoys is performed in rotation on each of them, in this case the workload will be fair but the maximization of resource consumption at a buoy is not considered.

### 5.2.1 Function to assign vessel requests to buoys using round robin

Below is the code that allows you to perform a round robin assignment within each AS:

***def assign\_requests\_round\_robin(x1, x2, u1, u2, k, num\_vessels, min\_energy, max\_energy, max\_water, last\_assigned\_buoy):***

***for i in range(num\_vessels):***

***assigned = False***

***for \_ in range(n):***

***j = (last\_assigned\_buoy + 1) % n***

***if (x1[j, k + 1] - u1[i, k] >= min\_energy and x1[j, k + 1] - u1[i, k] <= max\_energy and***

***x2[j, k + 1] - u2[i, k] >= 0 and x2[j, k + 1] - u2[i, k] <= max\_water):***

***# Assegna la richiesta alla boa j***

***x1[j, k + 1] -= u1[i, k]***

***x2[j, k + 1] -= u2[i, k]***

***if x1[j, k + 1] < min\_energy:***

***x1[j, k + 1] = min\_energy***

***if x2[j, k + 1] < 0:***

***x2[j, k + 1] = 0***

***print(f"Vessel {i + 1} assigned to Buoy {j + 1}:")***

***print(f" Energy requested: {u1[i, k]}")***

***print(f" Water requested: {u2[i, k]}")***

***print(f" Updated energy for Buoy {j + 1}: {x1[j, k + 1]}")***

***print(f" Updated water for Buoy {j + 1}: {x2[j, k + 1]}")***

***assigned = True***

***last\_assigned\_buoy = j***

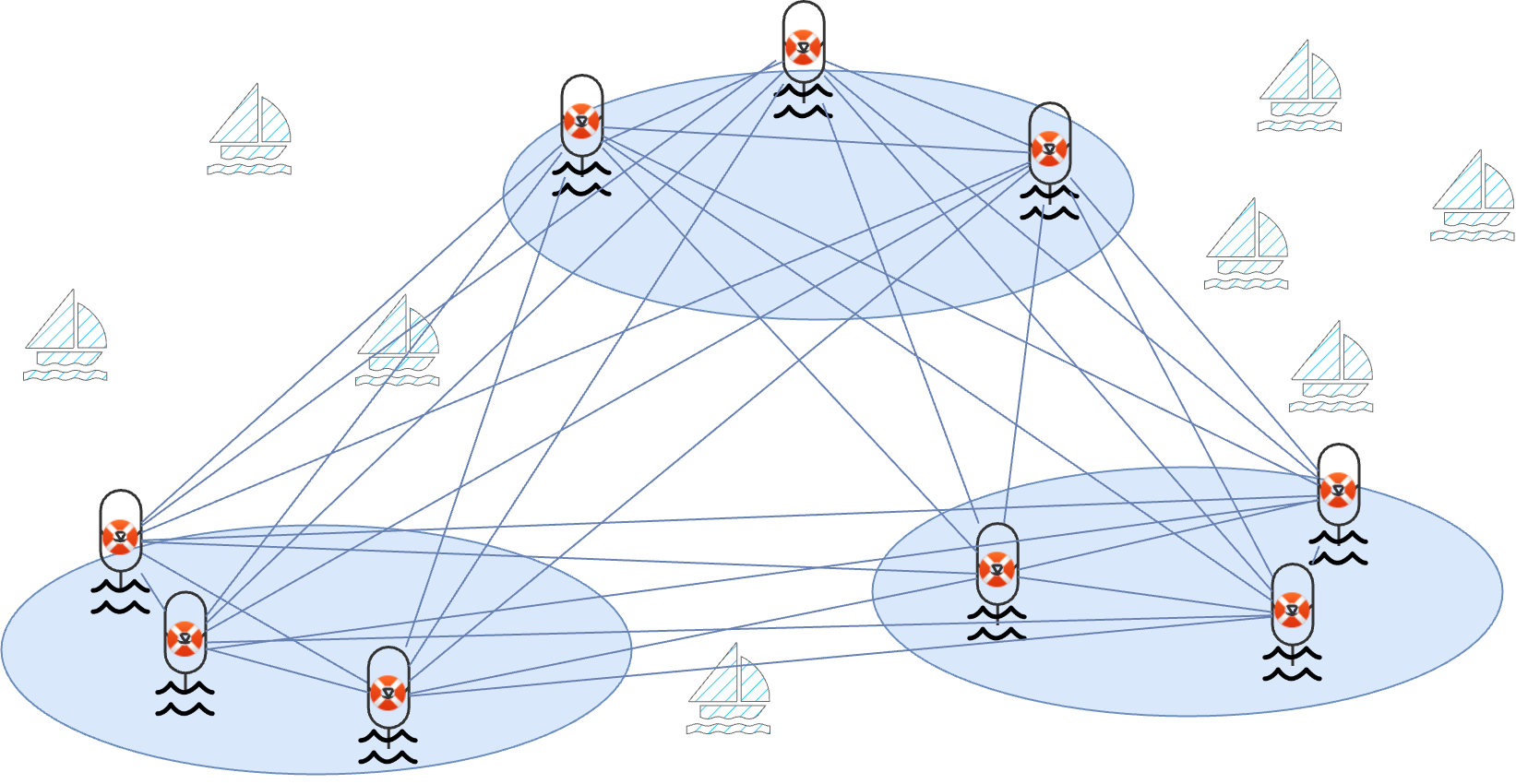
***break***

***if not assigned:***

***print(f"Vessel {i + 1} could not be assigned at time step {k + 1}")***

***return last\_assigned\_buoy***

## 5.3 Greedy method



In the given context, the **Optimized Greedy Assignment** algorithm is applied to manage the allocation of energy and water resources from autonomous buoy systems to incoming vessels. Here's an explanation of how the algorithm fits into this application, along with its advantages and disadvantages.

***Context of Application***

Scenario

The system consists of multiple autonomous buoy systems, each with a set number of buoys. These buoys accumulate energy from renewable sources and store water. The buoys need to provide these resources to incoming vessels, which arrive at various times and have different energy and water demands.

Goals

1. **Efficient Resource Management**: Ensure that the buoys can effectively meet the demands of the vessels without depleting their resources below critical levels.
2. **Balanced Load Distribution**: Prevent any single buoy from being overburdened, thus ensuring the longevity and reliability of the buoy system.

Advantages of Optimized Greedy Assignment

1. **Simplicity and Ease of Implementation**:
   * The greedy algorithm is straightforward to understand and implement, making it suitable for real-time applications where quick decision-making is essential.
2. **Fair Resource Allocation**:
   * By choosing the buoy that can best handle a request with minimal impact, the algorithm helps distribute the load evenly across all buoys, preventing resource exhaustion in any single buoy.
3. **Scalability**:
   * The algorithm can easily scale with the number of buoys and vessels. Even as the system grows, the greedy approach remains manageable in terms of computational complexity.
4. **Local Optimization**:
   * The algorithm makes optimal local decisions, which can often lead to good overall system performance, particularly in dynamic environments where conditions change frequently.
5. **Real-Time Response**:
   * The greedy nature allows for quick decisions, which is critical in real-time systems where delays in allocation could lead to resource shortages or other operational issues.

Disadvantages of Optimized Greedy Assignment

1. **Global Optimality**:
   * The algorithm focuses on local optimization and does not guarantee a globally optimal solution. This means that while each individual decision is optimal, the overall resource allocation might not be the best possible.
2. **Sensitivity to Parameters**:
   * The effectiveness of the algorithm can be sensitive to the chosen parameters, such as the minimum energy threshold and the rate of energy and water production. Incorrect parameter settings can lead to suboptimal performance.
3. **Resource Depletion Risk**:
   * If the arrival rate of vessels or their resource demands are higher than expected, the buoys might deplete their resources faster than they can replenish, leading to potential service failures.
4. **Computational Overhead**:
   * Although generally efficient, the algorithm requires checking the impact on all buoys for each vessel request, which can become computationally intensive with a large number of buoys and requests.
5. **Lack of Long-Term Planning**:
   * The greedy approach does not consider future requests or changes in the system. This lack of foresight can lead to situations where resources are allocated inefficiently over the long term.

Summary

The **Optimized Greedy Assignment** algorithm provides a practical solution for managing resources in autonomous buoy systems, balancing simplicity and efficiency with fair resource allocation. However, its focus on local optimization and sensitivity to parameters mean that it may not always produce the best global results. In dynamic and real-time applications, it offers quick and often effective decision-making, but it requires careful tuning and may need supplementary strategies to handle extreme scenarios or long-term planning.

### 5.3.1 Function to assign vessel requests to buoy systems using an optimized greedy algorithm

Below is the code used to assign resources according to a greedy algorithm optimized within each AS:

***def assign\_requests\_optimized(system, system\_index, u1, u2, k, num\_vessels, min\_energy, max\_energy, max\_water):***

***x1, x2 = system***

***for i in range(num\_vessels):***

***best\_buoy = None***

***min\_impact = float('inf')***

***for j in range(n):***

***if (x1[j, k + 1] - u1[i, k] >= min\_energy and x2[j, k + 1] - u2[i, k] >= 0):***

***impact = (x1[j, k + 1] - u1[i, k]) + (x2[j, k + 1] - u2[i, k])***

***if impact < min\_impact:***

***min\_impact = impact***

***best\_buoy = j***

***if best\_buoy is not None:***

***x1[best\_buoy, k + 1] -= u1[i, k]***

***x2[best\_buoy, k + 1] -= u2[i, k]***

***if x1[best\_buoy, k + 1] < min\_energy:***

***x1[best\_buoy, k + 1] = min\_energy***

***if x2[best\_buoy, k + 1] < 0:***

***x2[best\_buoy, k + 1] = 0***

***print(f"Vessel {i + 1} assigned to Buoy {best\_buoy + 1} in System {system\_index + 1}:")***

***print(f" Energy requested: {u1[i, k]}")***

***print(f" Water requested: {u2[i, k]}")***

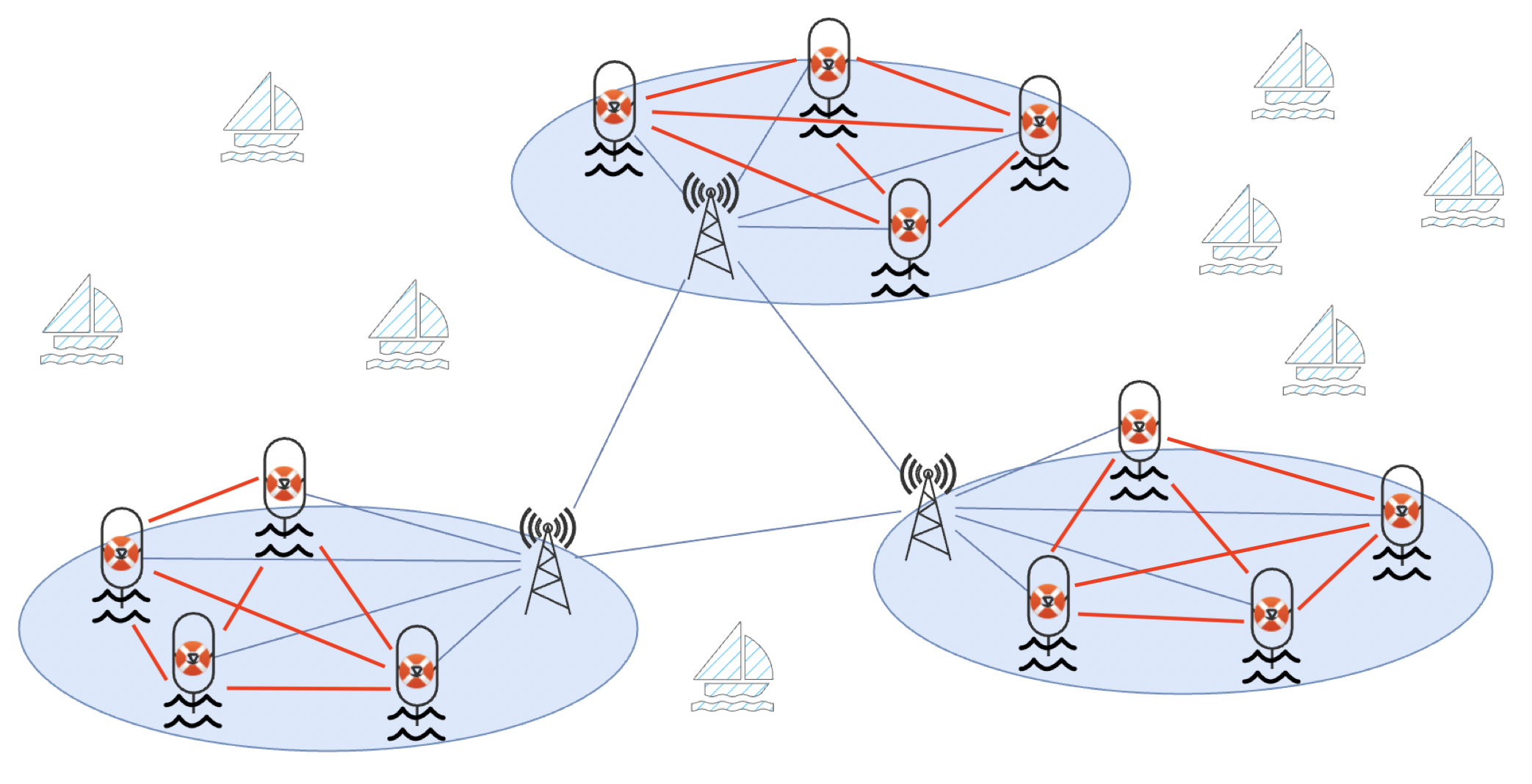
***print(f" Updated energy for Buoy {best\_buoy + 1} in System {system\_index + 1}: {x1[best\_buoy, k + 1]}")***

***print(f" Updated water for Buoy {best\_buoy + 1} in System {system\_index + 1}: {x2[best\_buoy, k + 1]}")***

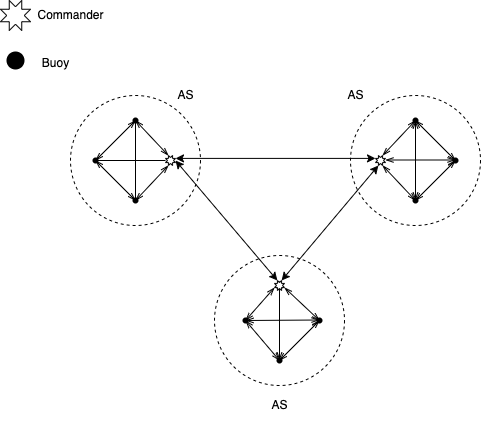
***else:***

***print(f"Vessel {i + 1} could not be assigned at time step {k + 1}")***

## 5.4 Problem of the Byzantine generals



This configuration results in a condensed graph distributed in the following way:



In our context the problem of the Byzantine generals has been used in two different contexts: between different ASs and within each AS. Each commander who may be a transmitter or a special buoy will be responsible for making decisions regarding the assignment of a vessel to a buoy or another AS. To choose the best possible version of the algorithm, it would be desirable to find information about transmission protocols, types of transmission, sources of interference, etc. Surely the normal version of the algorithm (which requires a number of commanders n equal to 3t+1 with t equal to the number of traitors is not the best possible solution, since the high availability of the data must also be guaranteed if a commander buoy is compromised and start sending incorrect data).

Description of the Byzantine Generals Problem

The Byzantine Generals Problem is a classic problem of distributed consensus first formulated by Leslie Lamport, Robert Shostak, and Marshall Pease in 1982. The problem is a metaphor for describing how distributed systems can reach a correct agreement even in the presence of malfunctioning or malicious nodes.

Problem Scenario

Imagine a group of Byzantine army generals who need to agree on a common strategy, either to attack or to retreat. The generals are in separate camps and can only communicate through messengers. Some of these generals might be traitors and try to confuse the others by sending conflicting information.

The objective is for all loyal generals to reach a consensus on the same decision (attack or retreat) and ensure that the traitors cannot cause the loyal generals to make different decisions.

Assumptions of the Problem

1. **Generals and Messengers**:
   * Each general can send messages to the other generals.
   * Messages can be intercepted, delayed, or falsified by the traitors.
2. **Loyalty and Betrayal**:
   * Some generals might be traitors and send false information.
   * Loyal generals must still reach a common agreement.

Solution to the Problem

The problem can be solved with distributed consensus protocols that ensure:

1. **Agreement**: All loyal generals agree on the same decision.
2. **Validity**: If all loyal generals propose the same decision, then that decision will be the final outcome.

Lamport, Shostak, and Pease's Algorithm

A well-known algorithm to solve this problem includes the following steps:

1. **Consensus Initiation**: Each general sends their proposal to all other generals.
2. **Message Collection**: Each general collects the proposals received from other generals.
3. **Message Relaying**: Each general forwards the information received to other generals for a sufficient number of rounds to ensure consensus.
4. **Final Decision**: Each general uses a majority function (or another aggregation function) to determine the final decision based on the messages received.

Application of the Byzantine Generals Problem in Autonomous Buoy Systems

In the simulation of autonomous buoys, the Byzantine Generals Problem is used to ensure that decisions regarding the distribution of resources (energy and water) are correct and consistent, even in the presence of erroneous or malicious information from some systems.

Operation in the Context of Autonomous Buoy Systems

1. **Information Gathering**:
   * Each Autonomous System (AS) gathers information about resources (energy and water) from the buoys of other AS.
2. **Information Verification**:
   * Each AS verifies the information received using a majority mechanism. This helps to filter out erroneous or malicious information.
3. **Consensus**:
   * Decisions about resources are made based on the verified information, ensuring that even if some AS are compromised, the final decision will be correct and based on the majority of reliable information.

Advantages and Disadvantages

**Advantages**:

* **Robustness**: Protects the system against failures and malicious attacks.
* **Consistency**: Ensures that all AS have a consistent view of the state of resources.
* **Reliability**: Increases the overall reliability of the system in dynamic and uncertain environments.
* **Scalability**: Can be applied to large distributed networks.

**Disadvantages**:

* **Complexity**: Increases the system's complexity, requiring sophisticated algorithms for consensus.
* **Delays**: Introduces delays in decision-making due to the need to gather and verify data.
* **Communication Overhead**: Requires substantial information exchange between AS, increasing network traffic.
* **Computational Costs**: Requires additional computational resources for processing and verifying data.

Conclusion

The use of the Byzantine Generals Problem in the context of autonomous buoys provides a robust and reliable solution to ensure correct resource distribution decisions, even in the presence of erroneous or malicious information. However, this robustness comes at the cost of increased complexity, delays, and communication overhead. The decision to implement such a mechanism depends on the importance of resilience and data consistency compared to operational costs.

### 5.4.1 Function to share and verify data between systems thanks to the Byzantine generals problem

Below is the code used for verifying and sharing data between ASes and within each of them in order to reach consensus on the assignment of the vessel's request to a buoy:

***def assign\_requests\_to\_buoys(system, u1, u2, k, num\_vessels, min\_energy, max\_energy, max\_water, system\_index):***

***x1, x2 = system***

***for i in range(num\_vessels):***

***best\_buoy = None***

***min\_impact = float('inf')***

***for j in range(n):***

***if (x1[j, k + 1] - u1[i, k] >= min\_energy and x2[j, k + 1] - u2[i, k] >= 0):***

***impact = (x1[j, k + 1] - u1[i, k]) + (x2[j, k + 1] - u2[i, k])***

***if impact < min\_impact:***

***min\_impact = impact***

***best\_buoy = j***

***if best\_buoy is not None:***

***x1[best\_buoy, k + 1] -= u1[i, k]***

***x2[best\_buoy, k + 1] -= u2[i, k]***

***if x1[best\_buoy, k + 1] < min\_energy:***

***x1[best\_buoy, k + 1] = min\_energy***

***if x2[best\_buoy, k + 1] < 0:***

***x2[best\_buoy, k + 1] = 0***

***print(f"Vessel {i + 1} assigned to Buoy {best\_buoy + 1} in System {system\_index + 1}:")***

***print(f" Energy requested: {u1[i, k]}")***

***print(f" Water requested: {u2[i, k]}")***

***print(f" Updated energy for Buoy {best\_buoy + 1} in System {system\_index + 1}: {x1[best\_buoy, k + 1]}")***

***print(f" Updated water for Buoy {best\_buoy + 1} in System {system\_index + 1}: {x2[best\_buoy, k + 1]}")***

***else:***

***print(f"Vessel {i + 1} could not be assigned to any Buoy in System {system\_index + 1}")***

***def assign\_requests\_to\_systems(systems, u1, u2, k, num\_vessels, min\_energy, max\_energy, max\_water):***

***for i in range(num\_vessels[k]):***

***assigned = False***

***for system\_index, system in enumerate(systems):***

***x1, x2 = system***

***for j in range(n):***

***if (x1[j, k + 1] - u1[i, k] >= min\_energy and x2[j, k + 1] - u2[i, k] >= 0):***

***print(f"Commander of System {system\_index + 1} is considering assignment of Vessel {i + 1} to Buoy {j + 1}")***

***assign\_requests\_to\_buoys(system, u1, u2, k, num\_vessels[k], min\_energy, max\_energy, max\_water, system\_index)***

***assigned = True***

***break***

***if assigned:***

***break***

***if not assigned:***

***print(f"Vessel {i + 1} could not be assigned to any System at time step {k + 1}")***

***def verify\_and\_consensus(systems, k):***

***print(f"Consensus process at time step {k + 1}")***

***for system\_index, system in enumerate(systems):***

***x1, x2 = system***

***energy\_reports = np.zeros((num\_autonomous\_systems, n))***

***water\_reports = np.zeros((num\_autonomous\_systems, n))***

***for other\_system\_index, other\_system in enumerate(systems):***

***if other\_system\_index != system\_index:***

***other\_x1, other\_x2 = other\_system***

***for i in range(n):***

***energy\_reports[other\_system\_index, i] = other\_x1[i, k + 1]***

***water\_reports[other\_system\_index, i] = other\_x2[i, k + 1]***

***verified\_energy = np.zeros(n)***

***verified\_water = np.zeros(n)***

***for i in range(n):***

***energy\_counts = Counter(energy\_reports[:, i])***

***water\_counts = Counter(water\_reports[:, i])***

***verified\_energy[i] = energy\_counts.most\_common(1)[0][0]***

***verified\_water[i] = water\_counts.most\_common(1)[0][0]***

***for i in range(n):***

***print(f"System {system\_index + 1} updating verified data for Buoy {i + 1}:")***

***print(f" Verified energy: {verified\_energy[i]}")***

***print(f" Verified water: {verified\_water[i]}")***

***x1[i, k + 1] = verified\_energy[i]***

***x2[i, k + 1] = verified\_water[i]***

# 6 Observations and Conclusions

In this report, we analyzed the issue of buoy assignment by considering various types of request allocations from different boats in multiple scenarios. We aimed to describe this issue as realistically as possible by modeling the random arrival of boats and distributing requests in a weighted manner over a 24-hour period.

We observed the convergence of resources (energy and water) according to different cycles of consumption and recharge at hourly intervals (each time step k equal to one hour). Additionally, we enhanced our code to maximize resource consumption on each buoy and increased the robustness of information exchange using the Byzantine Generals problem.

For future improvements, it would be beneficial to provide high availability in data exchange between autonomous systems (AS), potentially by increasing the number of generals within each AS to ensure greater robustness. Considering the geolocation of each boat is crucial; it would be impractical for a boat to be redirected to a distant buoy. The redirection should thus consider both the maximization of resource consumption per buoy and the proximity between buoy and boat.

Other important considerations include the information exchange protocols between buoys, boats, and AS, as well as the management of exceptional circumstances related to the technology used by the involved entities. The advantage of having AS with a smaller number of buoys lies in the faster convergence in decision-making compared to a fully distributed system (complete graph) with the constraint of distances between AS.

Each AS should be viewed as a unique context that communicates with other AS to provide the best possible service to the boat, as requests may randomly reach different AS. By adjusting the parameters of the models developed in the simulations, we can stress test the system in an ideal environment to understand the maximum load between resources and requests. This will ensure the evolution of the model and its development in a real-world context.